FIT3164 Data Science

Software Project 2

Test Report

Team	DS-02
Project title	Patient medical history summarisation
Team members	Rui Guo (30481872)
	Govind Chadha (31113567)
	Hamza Umar (31181465)

Introduction

The objective of this report is to create a testing report for our software which uses LLM's to provide us with summaries of medical data provided to us. This software is being developed to help the medical professionals who are time poor to access the patient's past medical history and provide a better diagnosis. Comprehensive testing is essential to verify the correctness, performance, and usability of both the model and the user interface. The Testing strategy created for our project was to test the code and other components of the software as we are creating them and then finally have all the tested and fixed components in one place.

Description of test approach used

Our testing plan mostly consists of manual testing done alongside different stages of software development. The testing of the software was added as a part of our sprints as it was easier for our team to test the implementation of the code while writing it so that we don't encounter errors before putting it into the final notebook.

1. The testing plan focuses to make sure that the generated summaries are human-readable and free of vague data.

2. The tool's functionality was tested for accuracy and clarity, with particular attention to the data cleaning process, making sure it prepares the data for LLM input.

3. Manual testing for the backend was conducted using VS Code and a smaller modified version of the original MIMIC discharge CSV file. For the frontend part testing was done by putting random inputs into the UI.

4. Tests involved random patients with different sets of medical conditions, visit numbers, and data lengths.

5. Important parts of testing included checking the quality of data cleaning on merged patient data, reformatting data into specific columns, and handling incomplete or missing data.

6. Edge cases, such as columns with single letters or no data, were also addressed. The summary quality will determine test success. No specific configurations, additional resources, or tools are required for this testing phase.

Test Results

Blackbox Testing 1:

We have written the black box testing report in accordance with the structure provided as part of our assignment.

a) What is being tested:

Using the black box testing on one of the major data pre-processing steps involved in our software, we tested the parse function of the software which is used for the reformatting of the text data given to us to create a new dataset. It will be cleaned and preprocessed further to be used as an input for the large language model (LLM).

b) How it is being tested:

The function was tested by conducting manual tests which included us running the code with an input with an expected output in mind, and when we receive the output we check if the actual output matches with the expected output. If it does, we can say that the black box test has passed.

```
def parse_text(text):
    # Split the text into parts based on the identified headings
    import re
    pattern = re.compile(r'(Name|Unit No|Admission Date|Discharge Date|Date of Birth|Sex|Service|Allergies|Attending|Chief Complaint|Major Surgical or Invasive Proce
    parts = pattern.split(text)[1:]  # Split and remove the first split since it's before the first heading

    # Create a dictionary to hold the extracted data
    data_dict = {}
    for i in range(0, len(parts)-1, 2):
        key = parts[i].strip()
        value = parts[i+1].strip()
        data_dict[key] = value

    return pd.Series(data_dict)

# Apply the function to the 'text' column
parsed_columns = df['text'].apply(parse_text)

# Concatenate the parsed columns with any other existing DataFrame columns
df_final = pd.concat([df, parsed_columns], axis=1).drop('text', axis=1)  # Optionally drop the original text column
```

Fig1.the code for parse_text function

c) What are the inputs to the code or part of the code being tested:

The input for the code is a dataset which is a random subset of the discharge csv in the MIMIC dataset.

	subject_id	text
0	10347477	\r\nName: ___ Unit No: _...
1	10347477	\r\nName: ___ Unit No: _...
2	10347477	\r\nName: ___ Unit No: _...
3	10347477	\r\nName: ___ Unit No: _...
4	10347477	\r\nName: ___ Unit No: _...
...	...	...
133	18203607	\r\nName: ___ Unit No: ___\r...
134	18203607	\r\nName: ___ Unit No: ___\r...
135	18203607	\r\nName: ___ Unit No: ___\r...
136	18203607	\r\nName: ___ Unit No: ___\r...
137	18886075	\r\nName: ___ Unit No: ...

138 rows × 2 columns

Fig2. input file containing data for random patients.

d) What are the expected outputs:

The expected output is a new dataset which is a re-formatted version of the text column of the input which is parsed into multiple columns alongside its subject id.

e) What are the actual outputs being observed:

The actual output is a new dataframe with subject id and text file parsed into multiple columns and it matches the expected output so the test passes.

...	...	...	...	...	...	...	...	...	...	...	...	...	
133	18203607	___	___	___	___	___	F	MEDICINE	Sulfa (Sulfonamide Antibiotics) / Nifedipine /...	___, ...	...	On admission:\r\n\r\n___ 11:00AM BLOOD WBC-10....	Ms. ___ is a year old fem with histor
134	18203607	___	___	___	___	___	F	MEDICINE	Sulfa (Sulfonamide Antibiotics) / Nifedipine /...	___, ...	...	Admission Labs:\r\n___\r\n\r\nWBC-13.4* RBC-3....	Assessment/Pl ___ yo female TAVR on
135	18203607	___	___	___	___	___	F	MEDICINE	Sulfa (Sulfonamide Antibiotics) / Nifedipine /...	___ ...	...	==========\r\n...	Ms. ___ is a year-old fem with a hist
136	18203607	___	___	___	___	___	F	MEDICINE	Sulfa (Sulfonamide Antibiotics) / Nifedipine /...	___, ...	...	___ 06:10PM BLOOD WBC-18.5* RBC-4.61 Hgb-13.3 ...	Pt was admit ___ abdominal pa Pt ha
137	18886075	___	___	___	___	___	M	MEDICINE	tramadol	___, ...	...	___ 02:05PM GLUCOSE-86 UREA N-11 CREAT-1.2 S...	Mr. ___ is a ___ man wit history of hyp

138 rows × 27 columns

Fig3. output file after running the function

Blackbox Testing 2:

We have written the black box testing report in accordance with the structure provided as part of our assignment.

a) What is being tested:

Using the black box testing we tested the application of the large language model (LLM) to our modified,merged and cleaned data. We then check the quality of the summaries produced and also check if the summaries have been integrated correctly back into the dataset.

b) How it is being tested:

The function was tested by conducting manual tests which included us running the code with an input with an expected output in mind, and when we receive the output we check if the actual output matches with the expected output. If it does, we can say that the black box test has passed.

```python
from transformers import pipeline

# Load the summarization model from Hugging Face's Transformers library
summarizer = pipeline("summarization", model="Falconsai/medical_summarization")

def summarize_text(text):
    """ Summarize the text using the loaded model, handling long texts by summarizing in parts if needed. """
    if len(text.strip()) == 0:
        return ""

    try:
        # Assuming the model can handle the entire text directly if not too long
        summary = summarizer(text, max_length=1024, min_length=150, do_sample=False)
        return summary[0]['summary_text']
    except Exception as e:
        print(f"Error summarizing text: {e}")
        return text

# List of columns to summarize
columns_to_summarize = [
    'Major Surgical or Invasive Procedure',
    'History of Present Illness',
    'Family History', 'Pertinent Results', 'Brief Hospital Course',
    'Discharge Diagnosis', 'Discharge Condition', 'Discharge Instructions'
]


# Apply summarization to each specified column and update the DataFrame directly
for column in columns_to_summarize:
    print(f"Summarizing {column}...")
    if merged_df[column].dtype == object:  # Ensure the column is of type object (textual data)
        merged_df[column] = merged_df[column].apply(lambda x: summarize_text(x) if pd.notna(x) else x)
```

Fig4. the code for summarise_text function

c) What are the inputs to the code or part of the code being tested:

The input to the code is a reformatted, cleaned and merged dataset containing the information of a patient in a csv format.

	subject_id	Sex	Service	Allergies	Attending	Chief Complaint	Major Surgical or Invasive Procedure	History of Present Illness	Past Medical History	Social History	...	Physical Exam	Pertinent Results	Bri Hospit Cour
0	10000032	F	MEDICINE	Percocet / Vicodin	___\n\n___.\n\n___\n\n___.	Worsening ABD distension and pain\n\nabdominal...	Paracentesis\n\n___ diagnostic paracentesis\r\...	___ HCV cirrhosis c/b ascites, hiv on ART, h/o...	- HCV Cirrhosis: genotype 3a \r\n- HIV: on HA...	___\n\n___\n\n___\n\n___	...	VS: 98.1 107/61 78 18 97RA \r\nGeneral: in NA...	___ 10:25PM GLUCOSE-109* UREA N-25* CREAT-0....	___ HC cirrhos c ascite hiv c AR h/c

1 rows × 21 columns

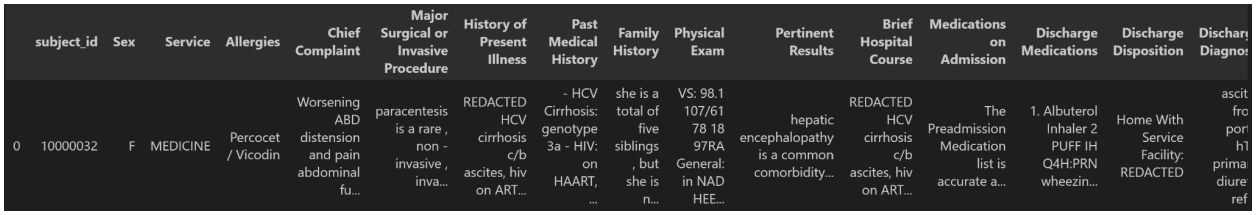
Fig5. the input for the summarise_text function

d) What are the expected outputs:

The expected output is a dataset with all the text data for a patient nicely summarized.

e) What are the actual outputs being observed:

The actual output is a dataset that contains the summaries of some columns from the input dataset reintegrated into the corresponding columns. Hence the test passes.

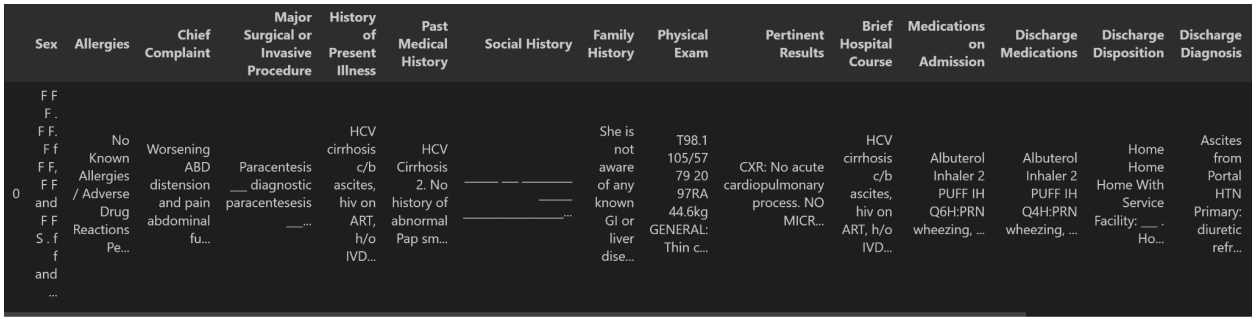


	subject_id	Sex	Service	Allergies	Chief Complaint	Major Surgical or Invasive Procedure	History of Present Illness	Past Medical History	Family History	Physical Exam	Pertinent Results	Brief Hospital Course	Medications on Admission	Discharge Medications	Discharge Disposition	Discharg Diagnos
0	10000032	F	MEDICINE	Percocet / Vicodin	Worsening ABD distension and pain abdominal fu...	paracentesis is a rare , non - invasive , inva...	REDACTED HCV cirrhosis c/b ascites, hiv on ART...	- HCV Cirrhosis: genotype 3a - HIV: on HAART, ...	she is a total of five siblings , but she is n...	VS: 98.1 107/61 78 18 97RA General: in NAD HEE...	hepatic encephalopathy is a common comorbidity...	REDACTED HCV cirrhosis c/b ascites, hiv on ART...	The Preadmission Medication list is accurate a...	1. Albuterol Inhaler 2 PUFF IH Q4H:PRN wheezin...	Home With Service Facility: REDACTED	ascit fro port h1 prima diure ref

Fig6. the output file of summarise_text function

f) Modifications required in code after running the test initially:

The actual data summaries received had quality issues for the columns containing very small texts in them (less than 2 lines).this problem was fixed by filtering the columns on which the large language model (LLM) was applied, and the quality of summaries increased after fixing the issue and provided us with the summaries that we expected.



	Sex	Allergies	Chief Complaint	Major Surgical or Invasive Procedure	History of Present Illness	Past Medical History	Social History	Family History	Physical Exam	Pertinent Results	Brief Hospital Course	Medications on Admission	Discharge Medications	Discharge Disposition	Discharge Diagnosis
0	F F F . F F. F f F F, F F and F F S . f f and ...	No Known Allergies / Adverse Drug Reactions Pe...	Worsening ABD distension and pain abdominal fu...	Paracentesis ___ diagnostic paracentesesis ___...	HCV cirrhosis c/b ascites, hiv on ART, h/o IVD...	HCV Cirrhosis 2. No history of abnormal Pap sm...	——— — ——— ——— ———————...	She is not aware of any known GI or liver dise...	T98.1 105/57 79 20 97RA 44.6kg GENERAL: Thin c...	CXR: No acute cardiopulmonary process. NO MICR...	HCV cirrhosis c/b ascites, hiv on ART, h/o IVD...	Albuterol Inhaler 2 PUFF IH Q6H:PRN wheezing, ...	Albuterol Inhaler 2 PUFF IH Q4H:PRN wheezing, ...	Home Home With Service Facility: ___ . Ho...	Ascites from Portal HTN Primary: diuretic refr...

Fig7. the previous output of the unmodified summarize text function

Integration Testing:

a) What is being tested;

The integration test is being conducted to verify the connections between the user interface and the backend data source. Although we do have an sql database but since it is not functionally connected, in this case our backend datasource is a csv file stored in a github repository. So

essentially, this test aims to check whether the UI can efficiently get, store and show the patient medical summary from the csv data file.

b) How it is being tested;

This test is conducted using python. The test basically starts the process of getting the data from the csv file and then displaying it based on the patient ID. The code shown reads the csv file from the github repository, matches the patient ID and checks if it is present in the dataset and then proceeds to show the patient medical summary in a table.

```
import pandas as pd

# Load the CSV data from GitHub
github_csv_url = "https://raw.githubusercontent.com/hamzaumar994/FIT3164/main/summary.csv"
data = pd.read_csv(github_csv_url)

# Display the first few rows of the DataFrame
data.head()
```

	subject_id	Sex	Service	Allergies	Attending	Chief Complaint	Major Surgical or Invasive Procedure	History of Present Illness	Past Medical History	Social History	Family History	Physical Exam	Pertinent Results	Brie Hospita Course
0	1	M	Cardiology	None	Dr. Smith	Chest pain	Coronary artery bypass graft	Patient presented with chest pain for 2 days	Hypertension	Non-smoker	Father had heart disease	Normal	Elevated troponin	Successfu surger
1	2	F	Neurology	Penicillin	Dr. Johnson	Headache	MRI brain	Patient presented with severe headache for 1 week	Migraine	Occasional alcohol use	Mother had migraines	Normal	MRI showed no abnormalities	Patien treated wit migrain medicatio

```
def fetch_patient_data(subject_id, data):
    patient_data = data[data['subject_id'] == subject_id]
    if not patient_data.empty:
        return patient_data.to_dict(orient='records')[0]
    else:
        return None

    from IPython.display import display, HTML

def display_patient_data(patient_data):
    if patient_data:
        html_content = '<table>'
        for key, value in patient_data.items():
            html_content += f'<tr><td><strong>{key}</strong></td><td>{value}</td></tr>'
        html_content += '</table>'
        display(HTML(html_content))
    else:
        display(HTML('<p class="error-message">No information found for this patient. Please check the Subject ID and try again.
```

Fig8 testing codes

c) What are the inputs to the code or part of the code being tested;

The inputs to the code being tested include the patient ID entered by the user in the user interface and the path to the github repository csv file containing patient data.

```
# Test with a valid subject_id
subject_id = 1
patient_data = fetch_patient_data(subject_id, data)
display_patient_data(patient_data)
```

Fig9 input

d) What are the expected outputs;

There are two main expected outputs of the integration test we run.
Firstly, when a valid patient ID is entered, the user interface is expected to display a summary of the patient's medical data in the form of a table extracted from the csv.
Secondly, when an invalid patient ID is entered, the user interface should display an error message depicting the absence of any information for the patient in the data.

e) What are the actual outputs being observed;

There are two cases for the outputs observed that depend on the type of input.
In case of valid patient ID: A table containing the patient summary

```
In [14]: # Test with a valid subject_id
         subject_id = 1
         patient_data = fetch_patient_data(subject_id, data)
         display_patient_data(patient_data)
```

subject_id	1
Sex	M
Service	Cardiology
Allergies	None
Attending	Dr. Smith
Chief Complaint	Chest pain
Major Surgical or Invasive Procedure	Coronary artery bypass graft
History of Present Illness	Patient presented with chest pain for 2 days
Past Medical History	Hypertension
Social History	Non-smoker
Family History	Father had heart disease
Physical Exam	Normal
Pertinent Results	Elevated troponin

In case of invalid patient ID: A message showing the correct error.

```
# Test with an invalid subject_id
subject_id = 33
patient_data = fetch_patient_data(subject_id, data)
display_patient_data(patient_data)
```

No information found for this patient. Please check the Subject ID and try again.

Fig10

Usability Testing:

a) What is being tested:

Usability refers to the ease of use and user friendliness of an application.
Usability includes several key aspects:
Learnability: how quickly users can learn to use the product to complete basic tasks.
Efficiency: How quickly users can complete tasks once they have mastered the system.
Memorability: how easily users can relearn how to use a system after not using it for a while.
Error rate: How often and how badly users make mistakes, and how easily they recover from them.
Satisfaction: How happy users are with the product.
Accessibility: It is often involved for specific user groups, ensuring that the product is friendly and usable for all.

b) How it is being tested:

In usability, we use user testing. After simple learning, users can use the application and give us feedback. This user test includes：learnability, efficiency, memorability, error rate, satisfaction, accessibility.

We use user ratings (1-10, 1 as worst, 10 as best)

c) Tester feedback:

Tester; Rui Guo

Questions	Score	Feedback
learnability	10	This application operation is relatively easy to get started

efficiency	8	This page is very efficient, as a developer there may be insufficient data update efficiency
memorability	10	This UI could be easy to memorize the operating steps.
error rate	9	If the subject id is not entered, filter can still be used
satisfaction	8	UI interface is clean but font size cannot be adjusted
accessibility	6	1. The default language cannot be changed; 2. There is no way for disabled people to use this page, such as audio reading, color blind mode, etc; 3. Font size and color cannot be adjusted;

Table1 usability test

Improvements and Limitations

Black box testing:

One of the limitations of the testing cases for the black box testing would be that the testing cannot check the quality of the summaries produced very well as we don't have the medical knowledge to judge the quality of summary produced so we go with the human readable summaries.

Integration testing:

Limitations:
Integration testing basically tests the overall working of the frontend and backend together. In this project case, one major limitation of the test is the failure in connection of the database to the user interface. Although we do have a SQL database, it is not connected to the user interface. The user interface is instead linked to a csv file with data in html which has restricted us to using very few patient cases. Due to this, our testing is largely based on a simulated environment and the results could differ if the backend is connected. Similarly, as the data is already manually entered, our testing model is not able to deal with real time data.

Improvements:
Functional Backend Integration:

1. Fully connected database and user interface to tackle live data
2. API endpoints implementation for operations on data

More Test Coverage:

1. Inclusion of further in-depth tests for edge cases where the patient id is missing or not available.
2. Further tests on vast amount of data for more accuracy

Automated Testing and Error Handling:

1. Use of multiple tools like Cypress or Selenium to deal with real time data and generate real user summaries.
2. Testing on cases for more scenarios where the system generates a useful message incase of error in data extraction from the database or any other backend error.

Usability testing:

Usability testing requires a large number of user test feedback. Colleagues should take into account that the target population of this application is doctors and patients, and we cannot obtain enough feedback in a short time to provide more perfect usability test results.

In the future project process, continue to do user interviews and user surveys, and timely obtain enough feedback information.

The tests not used:

The Pytest was not used in black box testing of our large language model because we are using a pre-trained LLM model and cannot have an exact same summary of the text every time the model is run to match with a predicted output.

Pytest was not used in parsing as well because we only need to match the column names as the data in columns is different for each patient and we cannot have a predicted output to match with for multiple patients.

Acknowledgement :

We acknowledge the use of ChatGPT(https://chat.openai.com/). Specifically, we utilize ChatGPT in the following ways:
1. Check our writing grammar.
2. The key aspects in usability.
3. For codes regarding integration testing.
4. Chat gpt was also used in the coding part of the software as a search engine, code debugging tool and a code helper too.